A Blog  
I’d like to use this blog as a paper trail of my curiosity from researching new and interesting things.   
Feel free to email me with any topics that you’d like to be discussed. I’m keen to answer any questions, and hear all thoughts/opinions.   
  
Hosting a Public Web Server from Home

3rd August 017

Contents:   
1. Intro

2. The Details

3. Conclusion

Intro

I’ve seen plenty of online portfolios. Especially from undergraduates studying Computer Science. To create my own, I embarked on a slightly more unconventional approach (in 2017 that is).   
  
The Raspberry Pi, an affordable, credit card sized computer was the instrument of choice. I’d bought one during university and it was just gathering dust. In an attempt save it from neglect, I used it to host this personal website.

Although the project was small and took less than a day in total (spent a few hours here and there when I found the time). It was an educational one.

Information consumed at university is often not given appropriate context.   
Depending on the module taken, you may find that much of the examinable material is theoretical. While this is beneficial for many reasons, it tends to commonly raises barriers when one tries to apply their newfound knowledge in a practical setting.

The main goal of this side project was to solidify familiar concepts by building something of interest.

Equipment:  
- MacBook Pro 2.4GHz Intel Core i5 16GB RAM - macOS Sierra  
- Raspberry Pi Model B 3

- TalkTalk HUAWEI HG633  
- 32GB MicroSD Card/Reader

- Micro USB cable   
- Ethernet Cable

The Details

In the interest of brevity, the focus of this post will be mainly on the lesser known concepts that you’re likely to encounter. However, I’ll still provide enough detail so that you could try the same at home. It’s also worth mentioning, my intention is not to explain in depth how these technologies work (as I’m still learning), but to instead give the reader a working knowledge to allow them to understand why and how things are done this way.   
  
1. Purchase a Domain Name  
Evidently, I bought the domain ‘blainemalone.com’. It cost £14.89 from ‘godaddy.com’. I’d advise that you do this at the end of the project if you are unsure that you’ll see it through (in the meantime use the routers IP address to test if website is live). Also, there are other domain registrars. Price around to find the best offers for you. With this said, I did appreciate GoDaddy’s intuitive DNS management dashboard and troubleshooting forums.   
  
2. Set up Raspberry Pi  
Insert your Micro SD card into the reader and then into your laptop/desktop computer. Now we want to download the Raspbian OS from here <https://www.raspberrypi.org/downloads/raspbian/> (select minimal image, only using Pi as a webserver. As a result, we don’t really require a GUI).   
Once downloaded, I used Etcher to burn the .img file to the SD card https://etcher.io/. For more information on how to do this, see <https://www.raspberrypi.org/documentation/installation/installing-images/README.md>.

After you completed this successfully, insert the SD card into the Raspberry Pi, connect the Micro USB for power and attach the Ethernet cable to both the Pi and the router. There are a series of common steps to setup a Raspberry Pi for use as a web server. For example, please ensure you have SSH enabled before moving on.

3. DHCP  
Once you have setup your Pi, you’ll want to know its IP address. In short, we want to be able to consistently locate our Pi in the local network.   
The best way to do this is to take advantage of the DHCP (Dynamic Host Configuration Protocol) Server built into your router. In short, DHCP helps with the allocation of IP addresses within a network. It removes the tedious process of manually configuring each machine which can be error-prone. As a brief reminder, every computer once it boots up, has a link layer address (MAC) embedded into their NIC (Network Interface Card). At this point it will not have an IP address so it must send a DHCP DISCOVER packet to the DHCP Server (Could be in same device as router or on a different network). In reply, the DHCP Server will allocate any one of its free IP addresses to the request using a DHCP OFFER packet. To be able to easily communicate with our Pi, we will want to ensure that the device is given a static IP address, i.e. one which doesn’t change, regardless of the device rebooting or loosing connection.

To achieve this using the TalkTalk HUAWEI HG633 router, I logged into the admin portal at <http://192.168.1.1/>. Under ‘Home Network’>’LAN Interface’>’DHCP Reservation’. This allowed me to map the Pi’s MAC address to an available IP address. So, anytime the DHCP Server receives a DHCP DISCOVER packet from the Pi’s MAC address it will provision the same IP address to it every time (e.g. 192.168.1.222). This task is generally router specific, so you should spend a little time getting used to your setup.

You can find out more about DHCP here.

4. Port Forwarding/The DMZ  
Usually you’ll want your personal website to be public facing, allowing anyone to view it. To accomplish this, you have two options.  
The first approach, is to make use of the port forwarding capabilities that come with your router. Port forwarding maps an exposed port on your WAN IP address to a port on a LAN computer. In other words, it is a way of making a computer on your home network accessible to computers on the internet, even though they are behind a router.   
For help setting up port forwarding, you can check out this website(<https://portforward.com/router.htm>). They’ve thoroughly covered many of the main manufacturers steps to configure port forwarding. Opting for the approach of placing your Pi behind the router will decrease your vulnerability to attacks. This is based on the assumption that your router doesn’t have a known exploit.

Alternatively, you can configure a DMZ (Demilitarised Zone), effectively placing your Pi on a separate LAN to your home network.   
In the often-likely event that a malicious party gains, or tries to gain, access to your server < https://security.stackexchange.com/questions/85232/lots-of-ssh-tries-on-my-log-is-it-normal>, he/she should be isolated in the DMZ network and not have direct access to the private hosts. For home routers, when enabling a DMZ containing a device, you must ensure that it’ll have a static IP. Otherwise a different device may inherit the designated IP address and become the DMZ host instead (not cool).  
  
Favouring simplicity, I opted for the DMZ. My reasoning was, if the Pi was compromised, I’d be caught between a rock and a hard place. The key was realising I’d be more comfortable with my Pi being hacked than my whole network being exposed. Clearly, I’m avoiding the rabbit hole of sophisticated network security. It’s an area that I’m interested in and intend to concentrate more on in later posts.

For now, here’s a few tips to quickly harden your webservers security:   
  
- Change the default username and password that is shipped with the Raspberry Pi. i.e. username: pi password: raspberry.   
- Change the default port 22 on the Pi’s SSH Server. This is to help avoid brute force attacks and keeps the log file cleaner. Choose a vacant port below 1024, as only root users can listen on ports below 1024 on most Unix-like systems.   
- Use fail2ban <https://www.fail2ban.org/> to prevent multiple attempts at an attacker guessing your password if they happen to find which port your SSH is listening on via scanning.

5. SSH & SFTP  
  
Using SSH (Secure Shell) to communicate with the Pi provides a secure medium to pass commands to the shell. Windows users can install ‘Putty’ SSH or Tera Term but Mac OS X users should already have SSH capabilities preinstalled.   
Using the static IP address configured earlier (192.168.1.222), connect to your device via the CLI from your terminal.   
Once authenticated, the next step is to install a web server. Initially I went with a traditional apache instance. However, after playing with glitch.com I gathered a little inspiration to mix things up. I settled on coupling Node.js with Express, this would become my web server.

Installing Node.js on Raspberry Pi  
After realising that installing Node.js on the Raspberry Pi wasn’t a walk in the park. I continued my research to later find a very convenient project dubbed node \_arm(http://node-arm.herokuapp.com/). After two simple steps, you’ll have it installed, saving hours of scratching your head.  
wget <http://node-arm.herokuapp.com/node_latest_armhf.deb>  
sudo dpkg -i node\_latest\_armhf.deb  
Make sure to install the Express framework once this is complete.   
npm install express --save

Next, you’ll want to test that everything is working as expected. There is no shortage of Node.js tutorials online (https://www.tutorialspoint.com/nodejs/nodejs\_express\_framework.htm)  
so take advantage of these to get up and running fast.   
If you’ve developed and executed the web app locally, it’s time to test it on the Pi. I found that the easiest way to manage files between your Pi and local machine is to use SFTP (SSH File Transfer Protocol)( <https://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol)>. Cyberduck is a free (when you download on their website, a download on the app store costs £22.99, I’m still not sure why) program that establishes SFTP connections with remote devices (<https://cyberduck.io/?l=en)>. Additionally, Cyberduck has distributions for Mac and Windows operating systems so it was an easy choice in this instance.  
  
<<< INSERT Cyberduck\_config.png HERE!!>>>

If you’ve followed this far, then congrats; both on understanding my vague instructions and finally being able to access your website publically.   
You should, in theory be able to access your sample website from your routers IP address.   
If not, double check that you’ve made sure that your Node.js web server is attached to port 80.  
<<app.listen(80);>>

6. DDNS (Dynamic Domain Name Service)  
Section 3 introduced DHCP, and how it is used within your LAN at home. More often than not, your home network/router will be assigned an IP address in the same way via the Internet Service Providers (ISP) DHCP server. Some business owners may pay premium to be allocated a static IP address by their ISP. If this is you, then happy days. Otherwise, you’re going to own a dynamic IP. This can cause a few problems if you want to host a website from home. Those of which can be mainly addressed with DDNS.

Firstly, what is DDNS?

Having a dynamic IP address makes it difficult for users to access their home network. Requiring them to know their new IP address every time it changes. A DDNS service like No-IP(https://www.noip.com/) will add a level of indirection by allowing your networks dynamic IP address to be hidden behind an easy to remember domain name. From this point, DNS will work as normal. For example, No-IP will issue you with a domain name for your network (exampleuser.ddns.net). The IP address associated with this domain name will be automatically updated by No-IP client side software when your ISP changes your IP address. Therefore, always referencing the domain name from No-IP allows your end users to seamlessly connect to the network without keeping track of the dynamic IP.   
But I’ve purchased a domain name, does this mean I can’t use it?   
No, you can simply redirect all requests from your purchased domain name to the No-IP domain name by a process called domain forwarding.

7. Containerization

Just hours after thinking everything was running smoothly, I’d realised my setup was far from perfect. I was sitting in a coffee shop when I noticed my site was down even though it had been working when I left the house. So, what could’ve went wrong? I sifted through quite a few reasons as to why it might have happened. Did my house get hit by lightning and kill the electricity supply? Did I get so many users that my Pi caved in on itself? Unlikely.   
It wasn’t until I got home until I realised why it wasn’t serving my content.   
My SSH session had timed out. The Node.js process had been executing in the foreground of the terminal, so when my session expired, the process died; taking with it all access to my website. I’d read about ways to automatically keep an SSH session alive from projects like <https://mistertea.github.io/EternalTCP/>. But I had a hunch that this wasn’t the road to go down. For me it seemed like a hack, as I didn’t want my node.js server to be attached to a terminal session. The solution? I ‘daemon’-ized the process, with Docker.  
From here, it was straightforward. I needed to run my Node.js server in a Docker container. Here’s how…

<<INSERT DOCKER BUILD FILE AND ALL COMMANDS>>  
  
By employing Docker into my solution, I’d discovered that it had a lot to offer. Amusingly, I’d realised that recovering from the aforementioned lightning storm would be easier than expected (assuming power was eventually restored). Adjusting my containers restart policy meant that anytime my container died, it would always restart itself.   
Raspberry Pi & Docker resources: <https://blog.alexellis.io/getting-started-with-docker-on-raspberry-pi/>   
<https://www.raspberrypi.org/blog/docker-comes-to-raspberry-pi/>

Conclusion

Would I recommend trying to host a personal website from home? Absolutely. It’s an enjoyable learning experience. However, I wouldn’t advise it for a long-term solution. At the time of writing, this site is currently hosted by AWS (<https://aws.amazon.com/)>. After noticing that my router suffered from intermittent outages for no apparent reason, robustness was a problem. I favoured ease of use and availability over anything else that the Pi had to offer.   
What are your thoughts on hosting a personal site from home? Send me an email with any questions/thoughts on this article.

<a href="mailto:blainemalone01@gmail.com">Contact me</a>

Thanks,

Blaine

\*\* Note: Please be careful when configuring your router at home. Make sure you understand the implications of what you are doing, otherwise you could leave your home network vulnerable. I’ve highlighted some of the obvious vulnerabilities. Please remember, even in the best case, your setup will only ever be as secure as the least secure portion of whatever is accessible over the Internet.